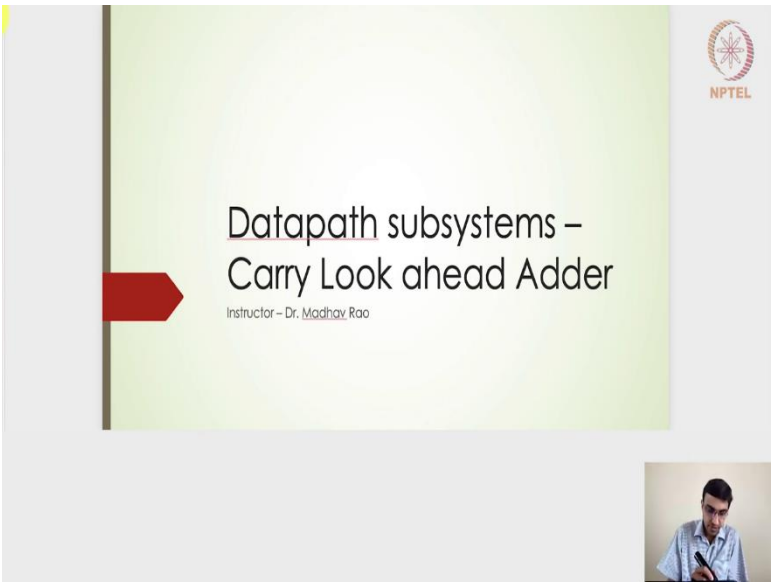
Design and Analysis of VLSI Subsystems
Dr. Madhav Rao
Department of Electronics and Communication Engineering
International Institute of Information Technology, Bangalore

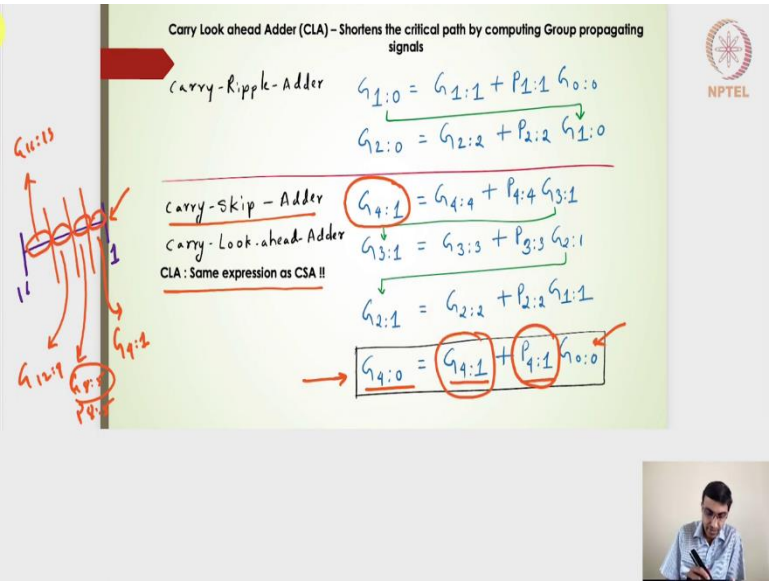
Datapath subsystems – Carry Look ahead Adder
Lecture - 90
Carry Look Ahead and Carry Increment Adder

(Refer Slide Time: 00:16)



Hello, students welcome to this lecture on the Carry Look ahead Adder, up till now we had seen the carry ripple adder and then the carry skip adder previously and then in this particular lecture we will take a look into the carry look ahead adder as well as just start or introduce you to the carry increment adder. Let us begin with the carry look ahead adder.

(Refer Slide Time: 00:39)



Let me also pick my pointer, the carry look ahead adder again it is very very similar to that of the carry skip adder. The carry skip adder we had this kind of an expression to go through that what it really means is we will make if I have a 16 bits starting from 1 bit to 16 bit.

We will make it into different groups here and in this particular case I have made 4 groups. Each of this groups we will create its own group generate, $G_{4:1}$ and this 1 will have a $G_{8:5}$ this 1 will have $G_{12:9}$ and then finally, this one will have $G_{16:13}$. Its individual subgroups or whatever the groups it is been formed it will create it is own group generate signals and those group generate signals are created simultaneously. Basically for $G_{16:13}$ it does not have to wait for any of the previous groups output.

Similarly, $G_{12:9}$ will be created simultaneously with respect to $G_{8:5}$ with respect to $G_{4:1}$. All of them will be available and then the moment $G_{0:0}$ is fed or it creates the $G_{4:0}$, that is what is written here.

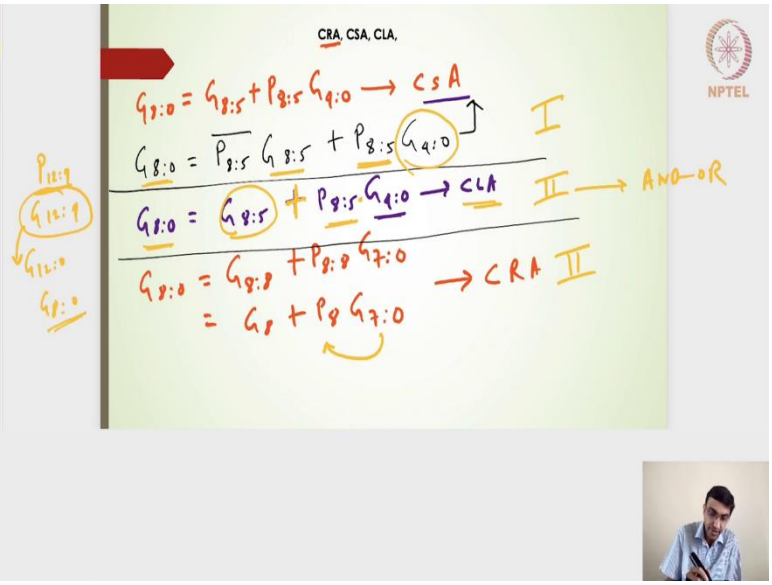
The $G_{4:0}$, this $G_{4:1}$ is created $P_{4:1}$ is created and then the moment $G_{0:0}$ is available it creates $G_{4:0}$. Similarly, whenever the $G_{4:0}$ is created we will get $G_{8:0}$, if we have $G_{8:5}$ and then $P_{8:5}$, that is what the carry skip adder was doing the additional carry skip adder does was it uses this particular expressions and then tries to deduce this expressions or implement this particular expressions in the form of a multiplexer.

It uses a multiplexer to skip some of the computation and based on the group propagate signals it actually skips the other computation and just passes into the next carry out, that

is about the carry skip adder. Let us have a look at the carry look ahead adder I have written the same expressions as that of the carry skip adder. Then what is actually the difference.

It also utilizes the same it makes it into the different groups of the bits and then each of the groups are responsible to create its own group generate bits and then the group or group propagate bits and group generate bits so that to get the carry out from that particular group.

(Refer Slide Time: 03:42)



It is very similar expressions we will have but we will see how it gets implemented. Let us take a look at the carry ripple adder. If I will write one particular expression

$$G_{8:0} = G_{8:5} + P_{8:5}G_{4:0}$$

This was done in carry skip ahead adder or carry skip adder in the carry ripple adder if I want to get $G_{8:0} = G_{8:8} + P_{8:8}G_{7:0}$ or $G_{8:0} = G_8 + P_8G_{7:0}$ and then it has to depend on the previous carry out coming from the 7th bit.

If I write this in terms of the bit level it will be $G_{8:0} = G_8 + P_8G_{7:0}$ as our carry ripple adder. The difference here is in the carry look ahead adder it will be the same thing as that of the carry skip adder. I am going to write the same expressions here $P_{8:5}$ and then $G_{4:0}$.

It also generates its own group generate signals and it is own group propagate signals and then it waits for the previous groups carry out signal. This is very similar to the carry skip adder, but in the carry skip adder we had actually used a multiplexer. If I write down the

expression, I am going to use a different color. The G 8 is to 0 was actually was the output coming from the multiplexers.

Then the multiplexer was something like this the select line was the group propagate signal and $P_{8:5}$, this was actually the carry skip ahead adder. I am going to demarcate this all these 3, so that we will get a clear cut idea. This is my the first one is the carry skip adder and then the second one is carry look ahead adder and the third one is our regular carry ripple adder.

Where the carryout gets rippled so that we will be able to generate the next carryout. The carry out keeps on rippling till the nth bit the carryout from the nth bit is generated. Whereas, in the carry skip adder it uses a multiplexer if the multiplexer what it really means is if I have $G_{8:5}$ is already available. The $G_{12:9}$ is already available $G_{16:13}$ is already available.

The moment $P_{8:5}$ if it is not 1 then it is already available and then it can directly if it is not one that means, if it is 0 then $G_{8:5}$ can be directly propagated as the carryout of $G_{8:0}$, that is the advantage the carry skip adder has.

Again, the $G_{8:0}$ is generated, but then again $G_{12:9}$ or rather $P_{12:9}$, if it is 0 then it whatever it has created $G_{12:9}$ it will be passed as the $G_{12:0}$, it actually skips the evaluation of the carry out here.

It actually skips the evaluation of $G_{8:0}$ need not have to compute because if $P_{12:9}$ is 0 then whatever is computed here $G_{12:9}$ it will be propagated as 12:0 as a carryout. It does not have to wait at all, but if it is 1 here then it has to wait for $G_{4:0}$ computation. So, that this CSA, the carry skip ahead adder has an advantage if the propagate group propagate bit is 0.

Whereas, this carry look ahead adder it does not use a multiplexers at all. It uses instead of deriving 1 inverter here and then putting it into a multiplexer it directly uses the AND gate here the AND gate and then here the OR gate. It uses the AND an OR gate, it uses the AND an OR gate instead of the multiplexers. The advantage is irrespective of the group propagate signals it is going to generate the $G_{8:0}$.

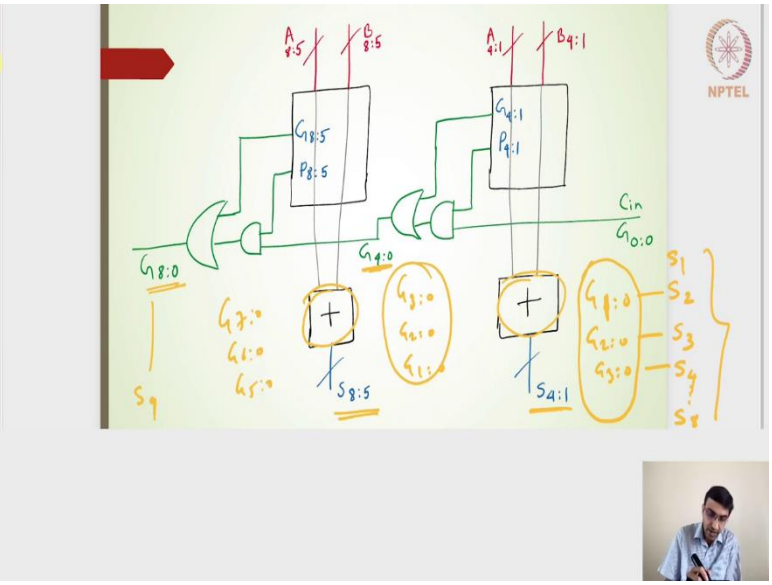
Even if in this particular case carry skip skip adder if $P_{8:5}$ is 1 then it has to do the AND computation and then that will be passed as $G_{8:0}$ and then the additional inverter delay is there for the $P_{8:5}$ here, that additional inverter is taken off and then it will do AND OR operation it has this is anyways made available. It will compute this $P_{8:5}$ this is also made available.

It will compute this AND operation and then do an OR operation and then get the carry out result which is nothing but $G_{8:0}$. It does not have to wait it does not have to do an inverter a this thing and whatever is when $P_{8:5}$ here is 1 it has to do an AND operation and then pass it as the $G_{8:0}$.

The passing through the multiplexers are generally done using the pass transistors and then it will have its own delay. In that sense it is a much more simpler design carry look ahead adder and we say that it is looking ahead the reason is very very simple, it does not really the moment 4:0 is generated we will get 8:0.

The moment 8:0 is generated we will get 12:0, the moment 12:0 is generated we will get 16:0. It does not it actually looks ahead to the carry out of the next group without even trying to calculate the 7:0 or 6:0 or 5:0.

(Refer Slide Time: 10:13)



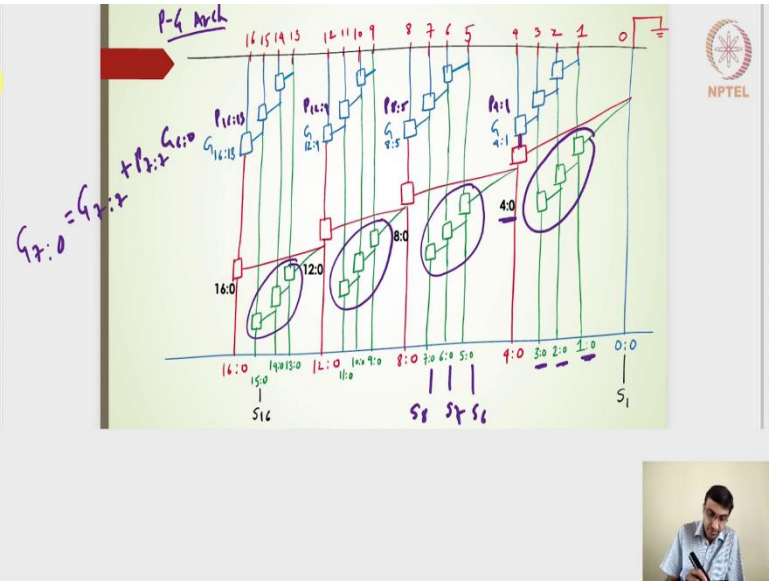
Moving ahead, the circuit or the block diagram of this particular subsystem, adder system is nothing but 2 blocks here and here. Which generates $G_{4:1}$, $P_{4:1}$ for the first group the

second group it generates 8:5, $P_{8:5}$ very very similar to that of the carry skip adder. Instead of the multiplexers here it will do an AND an OR operation, $P_{8:5}$ AND it with that of $G_{4:0}$ and then OR with that of $G_{8:5}$ and then it generates $G_{8:0}$. This is for only for the 8 bit addition we can extend 2 to more blocks for a 16 bit addition.

Remember that it ultimately it has to create the sum bits, ultimately it has to extract the sum bits not only S_8 or rather S_9 $G_{8:0}$ will give us the S9 sum bit, but we also want the sum bits from S_8, S_7, 6, 5, 4, 3, 2, 1. What it really has to do is internally in this particular adder it has to generate $G_{0:0}$, $G_{0:0}$ is anyways made available $G_{1:0}$, $G_{2:0}$, $G_{3:0}$.

After the XOR operation here it will give me S_2, S_3, S_4 and then so on. We want all the sum bits from 1 to 8. For to generate the sum 1 to S8 we need all this in the individual group generate bits. Even if I am taking the 1 to 4 bit this 1 is this particular logic is anyways is going to create 4:0, but still I need $G_{3:0}$, $G_{2:0}$, $G_{1:0}$, so that I should be able to calculate the other sum bits. This block is going to do that going to extract the $G_{3:0}$, $G_{2:0}$, $G_{1:0}$. The internal group generate signals this particular block is going to do similarly here this particular block is going to generate $G_{7:0}$, $G_{6:0}$, $G_{5:0}$.

(Refer Slide Time: 12:29)



The way it does is in the P G architecture, this is my PG architecture diagram alright. The first stage is nothing, but the bit wise the generate and propagate signals, the second stage is the group wise generate and propagate signals, the third one is nothing but having

implementing the XOR gate so as to extract the sum bits. We are considering the $G_{0:0}$ the carry input is nothing but grounded that is 0 object.

Notice here that all this blue box is generating its own group generate signals. The $G_{4:1}$ is created by this 3 this square block which is nothing but each of this blocks is representing AND an OR gate. It is creating $G_{4:1}$, this 1 is creating $G_{8:5}$, 12:9, 16:13, and simultaneously it is also creating $P_{4:1}$. It is not only $G_{4:1}$, but also $P_{4:1}$, it is also $P_{8:1}$, 8:5 this is also $P_{12:9}$, this is also $G_{16:13}$.

The output here is generating the group generate as well as the propagate signal. Once that is available then it goes to the instead of the multiplexers here it will go into the AND an OR gate. The red box is nothing but the bigger AND an OR gate which takes in $G_{4:1}$, $P_{4:1}$ and then $G_{0:0}$ so as to generate 4:0. Once the 4:0 is generated this is anyways made available along the same lines as when $G_{4:1}$, is the output generated.

The $G_{8:5}$, 12:9, 16:13 is all generated at the same time. Once we have that that will goes to the AND an OR gate the other input is coming from 4:0 and then the moment 4:0 is created it will try generating 8:0. The moment 8:0 is generated we will get 12:0, the moment 12:0 is generated we will get 16:0. This way we will be able to generate or look ahead into the carry out of the individual subgroups here.

Without even worrying about 7:0, 6:0 and 5:0 which are actually useful for generating this sum 8 bit sum 7 bit and sum 6th bit without even actually generating this it is actually generating 8:0. It is actually looking ahead to 16:0 without even considering when the 15:0, 14:0 or 13:0 is generate, and that is why it is called as the carry look ahead adder.

The other thing is once we understood these 3 blue boxes here for all the subgroups and then this particular red box to generate 16 is to 0 for a 16 bit addition. These 3 boxes which is nothing but a representing the AND an OR gates it is useful to generate 1:0, 2:0, 3:0 the moment 0:0 is made available.

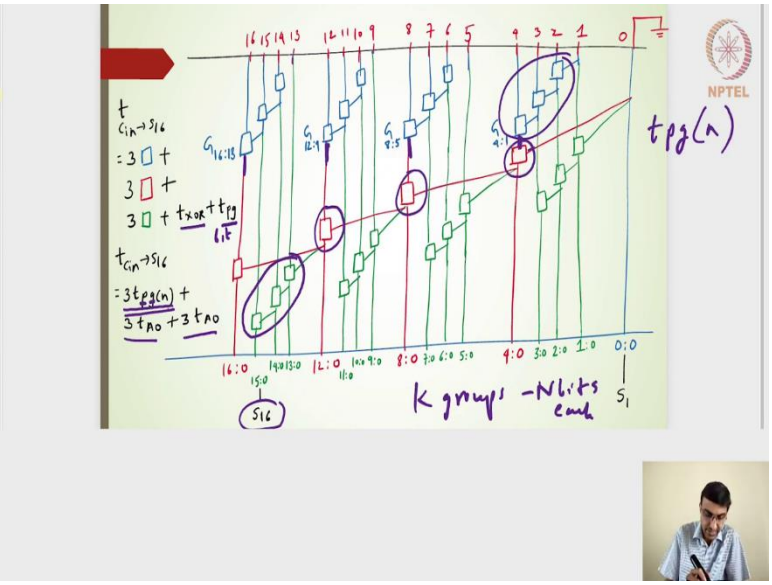
Similarly, the moment 4:0 is made available it will this particular block AND an OR gate is going to generate 5:0 this is going to generate 6:0, 7:0, and similarly 9:0, 10:0 and 12:0.

Now, remember that if I want to pick $G_{7:0}$. The $G_{7:0}$ is actually created by $G_{7:7} + P_{7:7} + P_{7:7}$ and it has to wait for $G_{6:0}$. Internally these 3 particular internally within that particular

group this is still a carry ripple addition only externally within interaction between the 4 groups it does utilize the carry look ahead adder, because here its own group generate and then propagate signals are created.

But, internally when it generates the 3:0, 2:0, 1:0 or 7:0, 6:0. 5:0 it actually does the carry ripple addition. Here also it is nothing, but the carry ripples, it actually waits for the previous one and then generates a $G_{4:1}$, $G_{8:5}$ $G_{12:9}$ 16:13. While it is interacting between the subgroups we have AND an OR gate so that we will get directly the 4:0, 8:0, 12:0, and 16:0 hope this is clear.

(Refer Slide Time: 17:04)



What should be the delay of this particular carry look ahead adder, it will have a bit wise generation delay it is nothing but an XOR gate AND gate whichever one is higher that we will have to account for.

In the third stage it will have an XOR gate delay, that is what we have the XOR gate and then the PG represents the bit wise this has to be bitwise PG generate propagate and generate signals. Then it will have you know these 3 blue boxes here the delay due to these 3 blue boxes the delay of this particular box. Then finally, S_{16} if I considering the 16 bit addition we are really interested in finding out the delay for generating these 16th bit sum. In that sense it will have the delay coming from these 3 particular blocks.

Overall, I will have the critical path coming from these 3 blue boxes these 3 red boxes and then these 3 green boxes. We will have this delay from the 3 blue boxes red boxes and then the green boxes. Now, what all these boxes are nothing but the AND an OR gate.

It is basically 3 AND an OR 3 and an or 3 AND an OR gates delay. The 3 and an or 3 AND an OR here and these particular blue boxes are written as nothing but PG of n. What it really means is it is been generalized into N bits having K groups of N bit each.

The N bits each, in that sense these 3 blue boxes these 3 blue boxes these 3 blue boxes 3 blue boxes can be generalized into the t_{pg} of n here where n here is actually nothing but the 4 bits.

It is basically generating the AND an OR gates at the output here this is what the output here whatever is the delay here that is considered as t_{pg} of n bits. A group wise propagate and generate signals for the n bits hope that is clear. We will have the delay as nothing but

$$t_{Sin \rightarrow S_{16}} = 3t_{pg(n)} + 3t_{AO} + 3t_{AO}$$

When I compare this with that of the carry ripple adder, we will have the same $3t_{ao}$ here the same $3t_{ao}$ here the only thing here is will be the multiplexers. The carry ripple adder will have the multiplexers here. The advantage with the carry skip adder is because of the multiplexers if the $P_{4:1}$ or $P_{8:5}$ or $P_{12:9}$ or 16:13 is actually 0 then it will actually take its own generated signal and then pass it to the next one.

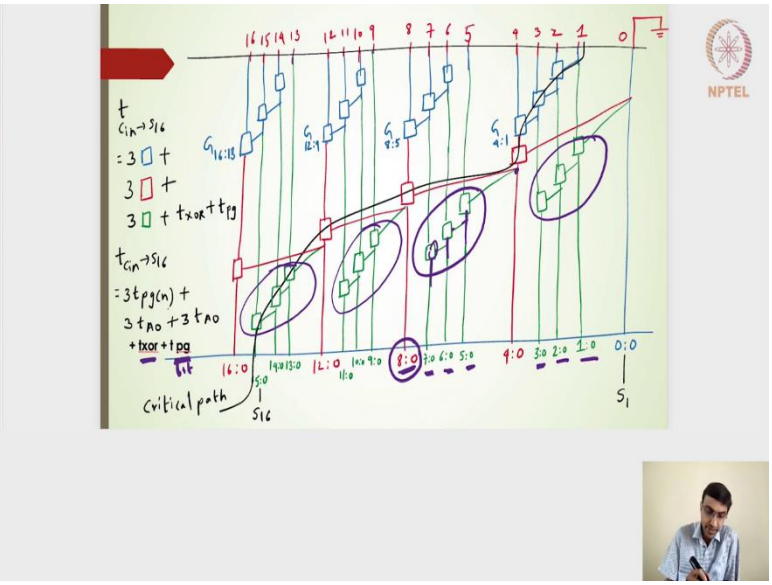
If this propagate group of signals is 0 this is 0 this is 0 then it will directly generate it will actually go really fast up till here and then it has to do only the 3 addition. In that sense it will be really faster it does not have to wait for this signal to come or rather this signal to be achieved.

It will be really fast to generate the output, but that will be one of the best case in terms of the worst case the $P_{4:1}$ or the $P_{8:5}$, $P_{12:9}$ and $P_{16:13}$ will be 1 and then it has to wait for the previous carry out signals.

In that sense I think the carry look ahead adder because it is a simple AND an OR gate does not involve any kind of an inverter to invert the group propagate signals we will still have a better performance or a delay here from the carry look ahead adder. That is why we

say that the multiplexers consume little bit more delay than that of a simple AND an OR gate hope this is clear to everyone.

(Refer Slide Time: 21:09)

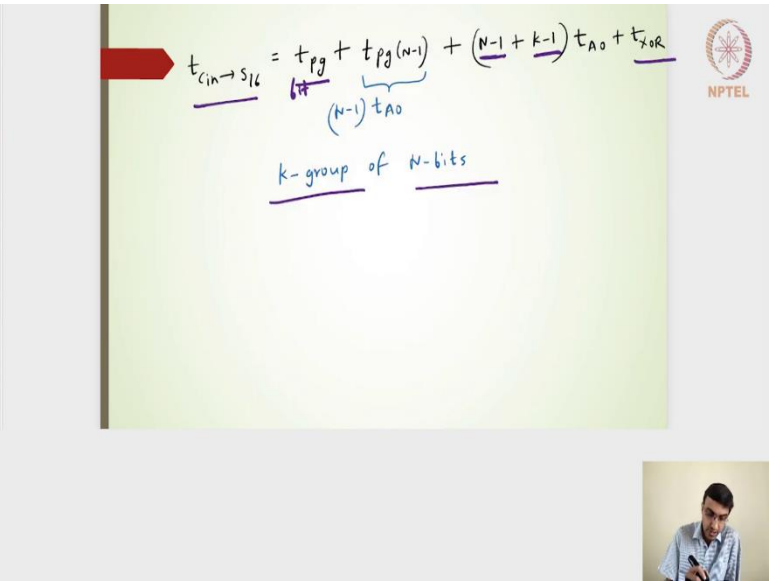


The overall delay or overall critical path is denoted like this by this particular black line indicating in this particular middle stage the critical path or the maximum delay that it takes. In this particular stage

$$t_{Cin \rightarrow S_{16}} = 3\square + 3\square + 3\square + t_{XOR} + t_{pg}$$

Itis nothing but the bit wise and then this t_{pg} of n is nothing but the group generate and propagate signals whatever is needed and whatever is the delay due to those particular AND an OR gates alright.

(Refer Slide Time: 21:45)



Hope this is clear, for a general expression of K group and N bits we will have

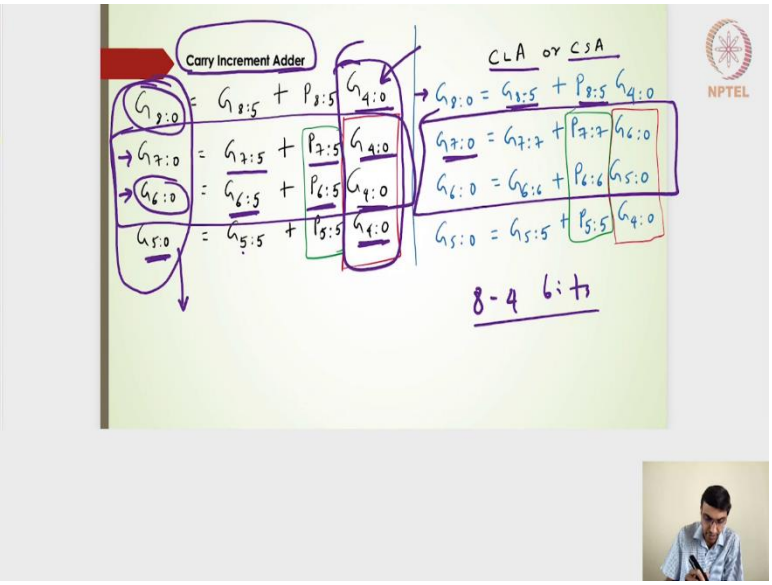
$$t_{Cin \to S_{16}} = t_{pg} + t_{pg}(N-1)(N-1+K-1)t_{AO} + t_{XOR}$$

This K is K - 1 decides the number of the red boxes which we had the AND an OR gates which is going to interface between the different subgroups.

This N - 1 is our last group's, the AND an OR gates and then this particular t_{pg} of N - 1 is our first AND an OR gates or the first groups AND an OR gates to generate the group propagate and generate signals. The XOR and will be nothing but for the last stage to extract the sum.

Then this t_{pg} which is nothing but bit wise is our whatever is the gates that has been involved whether it is an XOR gate or an AND gate whichever takes more to create the bit wise generate and propagate signals. That will be there in the first stage, hope this is clear.

(Refer Slide Time: 22:52)



Let us move on to the carry increment adder. The carry look ahead adder and carry skip adder the expressions were very similar or rather it was derived from this particular expressions. If I notice this $G_{8:0} = G_{8:5} + P_{8:5} G_{4:0}$. But if I look into the subsets of the carry out signals within the same group of 8 to 4 bits.

The $G_{5:0}$ is actually created from G5 plus or rather r with that of P5 and with that of $G_{4:0}$. The $G_{6:6} = G_6 + P_6 G_5$ and $G_{7:0} = G_7 + P_7 G_6$. If I look back into my PG level diagram here for the carry look ahead adder. I am talking about this particular 3 gates here 3 AND an OR gates 3 AND an OR gates 3 AND an OR gates here.

If I want to create the individual groups carry out and it subsequent the sum bits for example, 3:0, 2:0 or 1:0 or in this particular case 7:0, 6:0, 5:0 then I need this kind of a carry ripple kind of an architecture here. Where carry ripple in the sense it this one has to wait for $G_{4:0}$ here and then 5:0 is generated and this has to wait for 6:0 and then this has to wait for 6:0 and then 7:0 is generated.

Rather if it is a very sequential flow 7:0 has to depend on 6:0, 6:0 has to depend on 5:0. The question is can we actually design it in a sense that once the 4:0 is available why not all of them are made available 7:0, 6:0, 5:0 just like how 8:0 is made available. If I look into this particular expression of 8:0, 8:0 depends on this 4:0 and 8:5, $G_{8:5}$ and $P_{8:5}$.

Now similarly if I want to create the similar expression here this block can actually be brought into the same level, but I need not only 4:0, but also I will need 7:5, P_8 is $P_{7:5}$. Here for generating 6:0 at this level I will require $G_{6:5}$ and $P_{6:5}$.

That is what the carry increment adder does here. Here instead of waiting for $G_{6:0}$ to be generate in the 7:0 or 6:0 depends on 5:0 and 5:0 depends on 4:0 instead of that why not consider this 7:0 and 6:0.

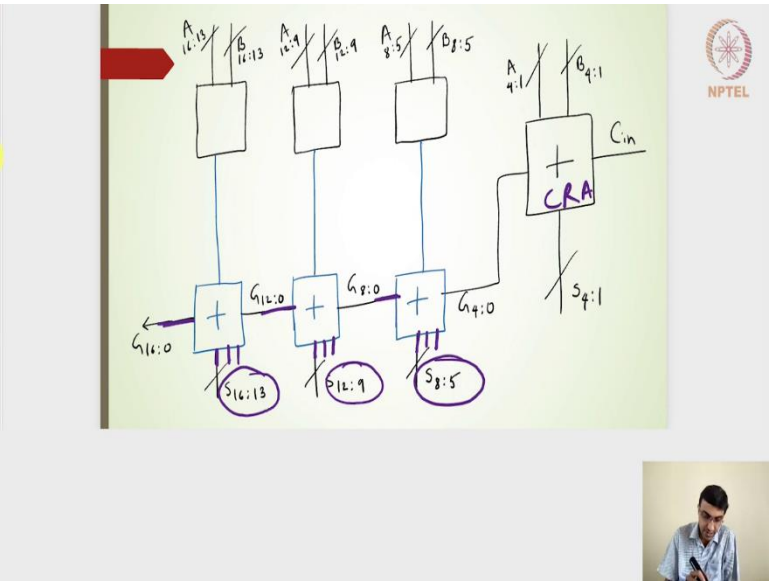
The 7:0 actually depends on 4:0 very very similar to 8:0, the way the 8:0 is generated it depends on 4:0 and then 8:5 and $P_{8:5}$. Similarly, if $G_{4:0}$ is generated and if 7:5 and $G_{7:5}$ is made available then with the help of $G_{4:0}$ it should be able to generate 7:0 at the same time as that of 8:0 and then 6:0 it depends on 4:0 it does not depend on 5:0 now and depends on fully on the 4:0.

The moment if it is because 6:5 and $P_{6:5}$ is made available the moment 4:0 is made available then we will get 6:0 at the same time as that of 8:0. The 5:0 is anyways depends on 4:0 and then 5:5 and $P_{5:5}$. If I look into the carry increment adder here these 2 particular are changing these 2 are changing with respect to these 2.

All other 2 things remains the same the carryout of the lower LSB side and then carryout of the MSB side in between the bits $G_{7:0}$, 6:0 there is it is being generated at the same time. The moment $G_{4:0}$ is made available here, all these things are created and it is waiting for $G_{4:0}$ is to be generated and then the moment $G_{4:0}$ generated all these 4 bits all these 4 group generate bits will be generated simultaneous.

In the sense each of this group 4, 4:1, 8:5, 12:9 and then 16:13 all this group is actually waiting for the carryout. The moment the carryout of the previous subgroups is generated it gets incremented, it gets holistically all the groups gets incremented into the group generate carryout bits. That is why it is called as a carry increment adder. It is waiting for one particular $G_{4:0}$ and that in the next instance we will get the $G_{8:0}$ to $G_{5:0}$ all of them will be in the next increment of the instance we will get all these carry out signals.

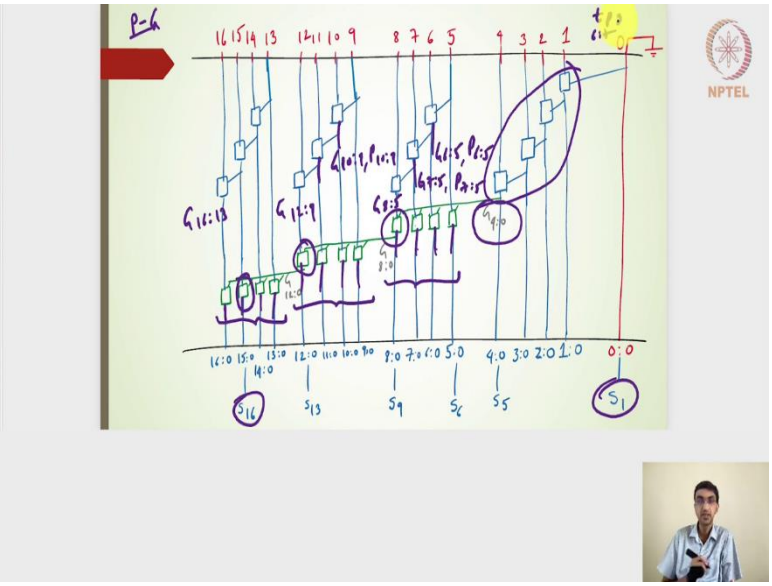
(Refer Slide Time: 28:15)



The block diagram is pretty simple you will have 4 groups here. This particular first group is 8 you know 4:1 and then this 1 is 8:5, 12:9, 16:13. The first group even if it does the carry ripple adder here. It is going to generate $G_{4:0}$ the moment $G_{4:0}$ is generated we will have $G_{8:0}$ that is been generated but also $G_{7:0}$, 6:0, 5:0.

The moment $G_{8:0}$ is generated we will have 12:0, but also we will have 11:0, 10:0, 9:0. The moment 12:0 is generated we will have 16:0, but also we will have 15:0, 14 and 13:0 which will help in extracting this sum bits.

(Refer Slide Time: 29:07)



The PG architecture here for the carry increment adder will look like something like this a carry ripple addition here in the first block. The moment $G_{4:0}$ is generated then it goes whereas while this particular 4 blocks is generating $G_{4:0}$ at the same time we will have 16 is to 13 being generated this one is 12:9 being generated $G_{8:5}$ being generated.

But not only along the outermost bit, but also on inside this particular bit we will have $G_{6:5}$ we will have $G_{7:5}$ and not only $G_{7:5}$ or $G_{6:5}$, but it will also have $P_{6:5}$ generated and $P_{7:5}$ generated and $P_{8:5}$ generated. Similarly here when it is generating this one it will be nothing but $P_{10:9}$.

Then similarly here $G_{11:9}$ and then $P_{11:9}$. The moment $G_{4:0}$ is generated I need one particular AND gate and an OR gate here which is represented by a block. To get my $G_{8:0}$, 7:0, 6:0 and then 5:0 simultaneously generated.

It is generated at the same time instance, it gets actually from $G_{4:0}$ it gets incremented to 8:0, 7:0, 6:0 and 5:0 at the same instance. Once 8 is to 0 is made available then it will get 12:0, 11:0, 10:0 and 9:0 incremented.

The moment 12:0 is made available then it will get 16:0, 15:0, 14:0 and then 13:0. In the sense the 4 groups which are being created it is actually once the $G_{4:0}$ is generated then 8:0 and all the other groups will be incremented in the next time instance. Then this will also be available in the next incremental time instance and then this will be available in the next incremental time instance. That is why it is called as a carry increment adder. Once we have this G_{16} in the individual carry out bits then I should be able to generate the S_{16} to S_1 bits. The critical path in this particular case will be nothing but the S_{16} bit how much amount of time it takes to generate the S_{16} bit and it will be nothing but coming from this particular AND an OR gate.

This the input here depends on this particular AND an OR gate and then that depends on this particular AND an OR gate this particular AND an OR gate. I will have this 4 AND an OR gates plus this 1 plus this 1 and then finally this 1.

Looking at it makes it very very simpler in fact, this is one of the most fast adder that we can develop the carry increment adder because the overall delay it turns out to be 4 t_{ao} and 3 t_{ao} to generate the $G_{15:0}$, and then one XOR to generate the S_{16} bit and then one the t_{pg} at the bit level pg at the bit level, that will be the delay.